Secure File Storage using Infrastructure of Encrypted Distributed Servers

Afifah Amjad[1], Ayesha Ali[2], Ayan Ahmar[3], Anas Ahmed[4], Abdul Rehman[5], Zeeshan Saleem Khan[6], Muhammad Wasim[7], Lubaid Ahmed[8]

***Abstract:* In today's digital age, data security has become a prime concern for individuals and organizations that rely on distributed systems for processing and storage. To overcome these issues, this research paper aims to address a new approach to data security, based on a distributed system that processes data in parallel. The purpose is to enhance the reliability of file storage and the security of the system and protect data from unauthorized access and data loss. Ensuring efficient access and retrieval of files, the system combines the two techniques; i.e., distributed storage and cryptographic techniques. The security of the system is ensured in a way that intruders cannot access the data until have access to all servers and encrypted keys. The proposed system operates as follows: when a user uploads a file, it breaks the file into different chunks, further, these chunks get encrypted. The key for each segment gets stored in the database. The segments will be stored on storage servers randomly. The record of the storage servers; that is where the data is stored is also maintained. When a user wants to retrieve a file, all segments from different servers compile to make up a single file, ensuring that the file cannot be retrieved until all of the segments have merged.**

***Keywords*: Data Segmentation, Encryption and Decryption, Distributed Systems, Parallel Processing, Vulnerabilities.**

INTRODUCTION

With the rise of digitalization, the dependency on technology grows. Therefore, cyber security has become a major concern. The rising number of cyber-attacks highlights the critical need to protect data security and integrity. The database management systems are at risk by threats of SQL injection, insider threats, and denial-of-service attacks [1]. Moreover cyber security in aviation industry needs continuous collaboration, information sharing to address emerging threats [2]. These attacks can be carried out during file transmission as well. Transfer of file with large data size persists great challenges [3].

At present, the most pressing problem is ensuring data security and integrity. In order to guard against cyber-attacks, strong encryption which provides high security of data should be implemented. An alternative security option that can be used to put data from ransom attack these days is by maintaining offline backup. Users shall, however, download files and be responsible for the safety of files on local storage. The more recent method to secure data is that before data goes onto the server, it is encrypted using AES (Advanced Encryption Standard) algorithm [4]. The same algorithm is used for the decryption of data. The principal goal of cryptography is to ensure that stored data is secure and kept away from cyber-attacks and any illegal activities.

Data partitioning eases secure storage and retrieval of data. Data segregation creates more flexible accessibility and lower costs for data storage. Another concept important for security is through dynamic operations, which secures the data at the time of storage on the server via encryption and decryption schemes. Assurance of data security in the provided spaces is performed by keeping and processing the data on same set of servers. Traditional security facilities such as firewalls and infiltration detection systems, no longer can guarantee security against cyber-attacks. For this reason, new and strong security facilities become mandatory.

This paper uses the well-developed concept of data segmentation [5]. Data segmentation is done by dividing large data into smaller chunks. These chunks are more manageable. This helps organizations to handle each chunk separately in a secure manner. This approach is particularly valuable in data management; as it helps in management of data into different servers in turns significantly improve the performance of the system. Data protection can also be maintained by dividing the data into the isolated segment. Organizations can apply safety measures on specific data chunks. If the specific data segment is compromised, the risk for other segments is very low. In addition, division is an effective strategy to meet regulatory requirements, such as general data protection regulation (GDPR) [6]. Sensitive information such as personal and financial is handled with a proper security level of safety. Data protection can be achieved by combining strategic data division and strong secure protocol. The purpose of this research paper is to develop a system that increases data security through strong encryption techniques on data segments and effective management of secured data chunks on distributed parallel servers.

[1-2-3-4-5-6-7-8] Department of Computer Science, Usman Institute of Technology
Karachi
Country: Pakistan
Email: * lahmed@uit.edu

BACKGROUND RESEARCH

Data is an important aspect of this modern world. Protecting it against both external and internal threats is a challenging task. Understanding the significance of data security is important because data security refers to a set of measures and guidelines for protecting sensitive information [6]. Data security protects data from unauthorized access and attacks.

This proposed system uses the concept of data segmentation and encryption. The proposed work allows users to upload their important files or data on the web application server that used local storage of machines for storing files and data. For uploading or getting access to the files, users are required to log into the application. The user can upload, delete, and download files. This web application prioritizes users' data security by employing a segmentation process and encrypting individual file segments. These encrypted segments are then distributed and uploaded onto different servers.

2.1 Encryption and Decryption

Cryptography is primarily concerned with ensuring the security of users' data. It aims to achieve several goals [7]. The primary goal of cryptography is data integrity. It ensures that data remain secure and unaltered, is kept safe from access by unauthorized parties, and is only accessible to parties with permission. Considering all objectives, cryptography addresses the challenges of data authenticity, integrity, and confidentiality. Information can be transmitted or sent using either an asymmetric or symmetric encryption algorithm [7]. Lacking the protection data of the users will directly threaten the privacy and security of users' data. To address privacy concerns, it is important to ensure proper isolation of data from other users. Additionally, data must be securely stored and protected [8].

The purpose of encryption is to maintain the confidentiality of the data from unauthorized modifications [9]. Symmetric and asymmetric encryptions are the two major types of encryption. Both methods ensure the encryption process is unique and cannot be easily replicated [9]. It is important to emphasize the importance of keeping the key secure and protected from unauthorized access. This is because the key is used to encrypt and decrypt the message, and if it is compromised, the message can be accessed by unauthorized individuals [10]. Symmetric keys utilize the same key for both the encryption and decryption processes. While in asymmetric encryption two keys are generated public and private keys. The private key is sent to the user secretly. The private key is used for the decryption.

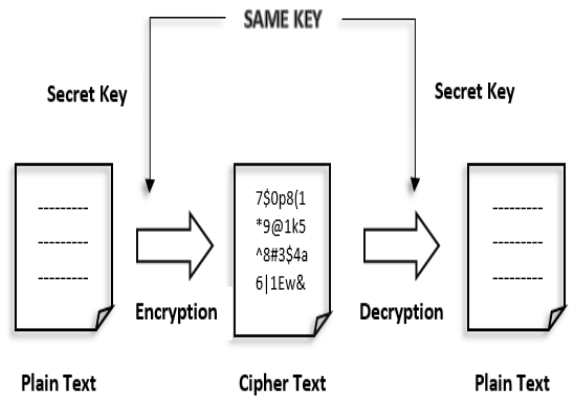

Figure 1: Symmetric Key Encryption

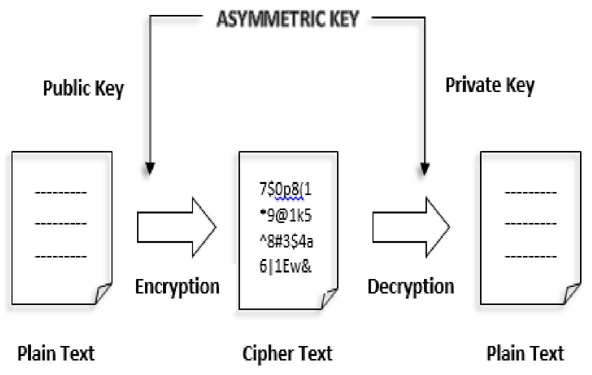

Figure 2: Asymmetric Key Encryption

2.2 Data Segmentation

Data Segmentation is the process of dividing data into relevant segments. Data segmentation involves categorizing and organizing information based on selected criteria, by breaking the data into smaller and manageable units. In the proposed system the segmentation technique is used for security purposes. The segments of the file will be stored on different random storage servers. This approach allows the simultaneous processing of data, with distributed chunks residing on various servers. Segmentation enhanced data security and faster processing speeds. This technique is useful because processing a big block is computationally slow rather than processing a small chunk. The other benefit is it ensures security because collecting all the chunks from different servers is more secure and complicated in it.

When the file gets uploaded to the system it gets segmented using the segmentation technique. Each of those segments gets encrypted using AES. The segments then get uploaded on the different servers randomly. Data partitioning is a technique used in database systems and big data analysis frameworks to improve performance by distributing data across multiple servers or nodes [11]. This allows for more efficient querying and processing of large amount of data.

The main objective of data partitioning is to enhance scalability and reduce the load on any one server this facilitate parallel processing and improves fault tolerance [11]. One way to achieve a more balanced distribution of data is by using techniques such as hash functions or range keys for partitioning. However, this approach can result in uneven partitions regarding data or computation, which may adversely affect the performance and lead to failures [12].

2.2.1 Fixed Size Segmentation Complexity

For the Fixed-type segmentation, its time complexity usually is O(n), where n is the number of chunks in which the file is split up. This occurs because the algorithm reads through the entire file as a whole, and it has to encrypt every chunk within about the same period of time. The fixed size segmentation usually be completed in linear time.

2.2.2 Variable Size Segmentation Complexity

For the Variable-type segmentation, time complexity is O(m*n), in which m represents the number of random numbers generated and n denotes the number of chunks that a file is split into. It happens because it reads the entire file in one go and encrypt every chunk in constant time and then generates random numbers inside a loop.

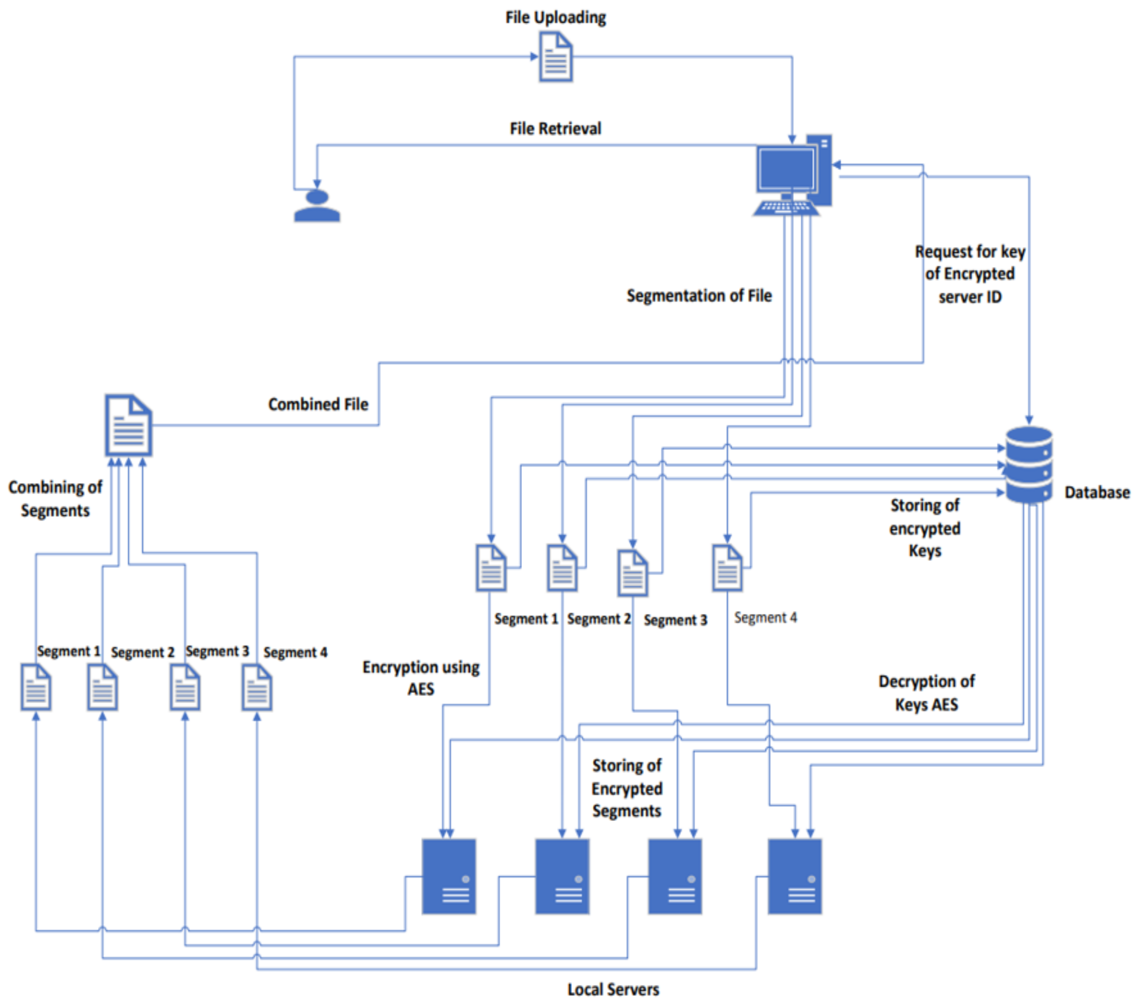

Figure 3: System Diagram

METHODOLOGY

3.1 Objectives

The primary goal is to develop a web-based application that can replicate various security-related services using encryption and decryption techniques. The system involves a software application that incorporates to ensure the protection of user information. The proposed system consists of the following main features.

- The system allows users to upload files.
- The user can access the system and download the file.
- The file can be removed from the system by users.

3.2 System Overview

The proposed system is based on local servers used for storing encrypted segments. The proposed system is in the form of a web application. The user or the admin can access the application by login onto the portal. When users logged in to the application user can upload or retrieve their files. When a user uploads the file, the file gets split into segments using a segmentation technique. These segments get encrypted and sent to the random servers, and these random server IDs are stored in the database to locate the segments at the time of retrieval. Stored IDs are in the encrypted form in the database. And if the user requests the retrieval of the file, the request goes to the database where it gets the encrypted keys to locate the server and decrypt the segments, then the request is sent to the servers where all the segments get combined and the file gets downloaded on the local storage.

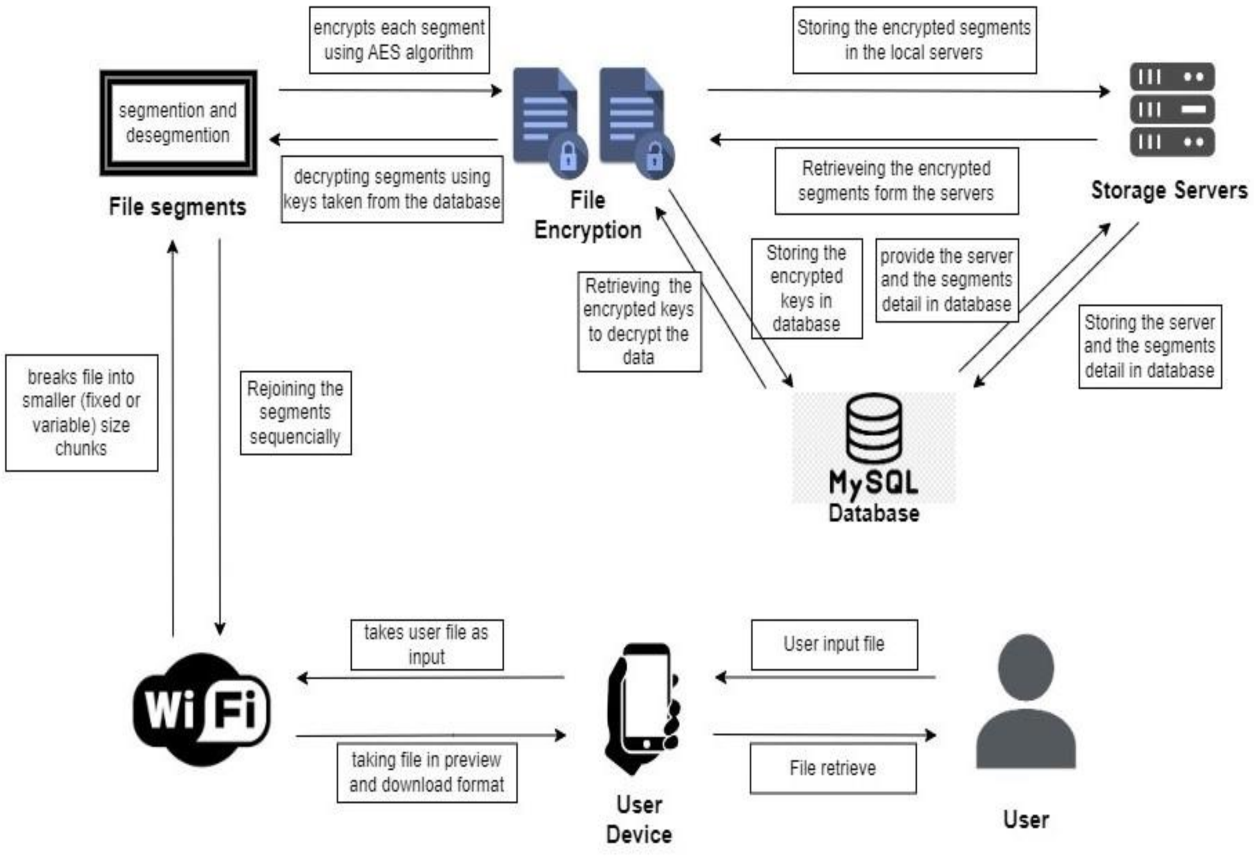
encrypts each segment using AES algorithm
segmention and desegmention
File segments
decrypting segments using keys taken from the database
File Encryption
Storing the encrypted segments in the local servers
Retrieveing the encrypted segments form the servers
Storage Servers
Storing the encrypted keys in database
provide the server and the segments detail in database
Retrieving the encrypted keys to decrypt the data
Storing the server and the segments detail in database
breaks file into smaller (fixed or variable) size chunks
Rejoining the segments sequencially
MySQL Database
takes user file as input
taking file in preview and download format
User input file
File retrieve
User Device
User

Figure 4: Operational Diagram of the System

3.3 File Server Functionality

Data and information need to be shared between employees and clients in any organization. Instead of using Universal Serial Bus drives or emailing to transfer data between computers. People can use a file server to serve as a central repository for their files. This feature can make it easier for people who are in various places to work together. A file server uses a computer as a server to store data and make it available to other clients on the network.

It serves as a central place to store data and share files. A server can be either confined to a single local area network or accessible through the Internet. To download uploaded saved files, access to the main system is required. When a file is selected for downloading, the system automatically retrieves the necessary key from the main server [13].

3.4 Binary Large Object

A BLOB, or binary large object, is a format of data storage within databases used for holding significant quantities of binary data, including images, videos, audios, and other forms of multimedia. Typically, BLOBs are stored as independent entities within the database rather than being part of a table row and can be accessed and manipulated using specialized database functions or APIs. Some DBMS support BLOBs natively, while others may require third-party software or plugins to manage them. In application development, there are several ways to store large objects. One approach is to use files in the file system, another is to save them as BLOBs in a database, and a third option is to combine both approaches [14]. In certain situations, a hybrid method may prove to be the most effective solution. However, it's worth noting that using MySQL tables for storage can lead to limitations in terms of table size, with some architecture only able to accommodate up to 8 terabytes per table [15].

3.5 AES Algorithm

Advanced Encryption Standard (AES) algorithm uses the symmetric technique [16]. The same key is used for encrypting and decrypting of the data. AES operates on data blocks of 128 bits and uses key lengths of 128, 192, and 256 bits for encryption and decryption [4]. The 128-bit data block is divided into 16 bytes, which are then organized into a four-by-four array referred to as the State. For each key, there are rounds defined; 128-bit keys use 10 rounds, 192-bit keys use 12 rounds and 256-bit keys uses 10 rounds. Encryptions consist of the following steps sub-bytes, shift rows, mix columns and round key.

The AES algorithm has undergone two modifications. These modifications include changes made to 1) the key schedule process and 2) the cypher round algorithm [17]. The decryption is performed by utilizing the reverse of the encryption that includes inverse shift rows, inverse substitute bytes, add round key and then inverse mix columns [4, 17]. A problem that may arise with the fixed-key approach is the risk of key-specific implementation being obtained and subsequently exploited in place of the actual key. This could allow an unauthorized individual to encrypt or decrypt any message for which the intended user has the capability to do so [18].

AES has replaced the DES in 2001 with new and updated features i.e., Block encryption implementation [19]. AES algorithm is more efficient because of the variety of key selection for encryption. AES allows you to choose in between 128, 192 or 256-bit keys, and make it stronger than the 56-bit key of DES.

Table 1: AES 128 vs AES 192 vs AES 256

	Key Length (Bits)	Block Size (Bits)	Rounds
AES 128	128	128	10
AES 192	192	128	12
AES 256	256	128	14

3.6 AES Working

The implementation of encryption algorithm is done using the mode of Advanced Encryption Standard with Galois Counter Mode (AES-GCM) [20]. AES-GCM is designed for higher performance of encryption and decryption. The reason for choosing GCM is to ensure both data authenticity and confidentiality within the system.

The implementation of AES utilizes predefined lookup tables [4]. Its primary functions include block cipher encryption with AES and performing multiplication over the matrix field through Galois Field Multiplication [19-21].

GCM serves as a mode of operation for AES that merges confidentiality with integrity. It encrypts the plaintext and generates an authentication tag, which can be used to confirm the integrity of the data. GCM is especially advantageous for real-time applications due to its ability to support efficient parallelization and pipelining. The rapid implementation of the GCM algorithm employs the traditional method of lookup tables for multiplication in a finite field. In the AES-GCM implementation, the key size is set at 128-bits, with the key size varying based on the required level of security. 128-bit of key length will cause the execution of 10 rounds. For each round all the steps for encryption will be performed that will include permutation and substitution.

AES performs encryption or decryption on a 128-bit block of plaintext or cipher text by applying the same round transformation multiple times, the number of which depends on the key size [4]. In AES-GCM, the process of encryption and decryption depends on the key length, the higher the key length it will take the more time. The more the number of rounds executed, the more time it will take to complete [20].

This method allows for a faster computation of the GCM algorithm [20]. The problem can be divided into four sub-problems, with each one containing four equations. In each sub problem, the same method described above is applied to each equation, to determine the most probable candidate for the 1 least significant bit of the lower-half key bytes and the 7 most significant bits of the single upper-half key byte, the problem is divided into smaller sub-problems, facilitating a more straightforward solution [22].

RESULT

Table 2: Testing result of multiple files

File Extension	Filename	File Size (MB)	Host Data	Uploading Time (sec)	Deleting Time (sec)	Downloading Time (sec)	Upload Average (sec)	Delete Average (sec)	Download Average (sec)
png	galaxy11.png	0.39	313	14	3	10	15.2	4.6	7.4
	face.png	0.13	2331	21	5	7			
	forgotpass.png	0.08	332	11	5	5			
	skeleton.png	0.07	133	20	6	6			
	database.png	0.06	113	10	4	9			
mp3	adelemusic.mp3	4.35	1133	37	14	15	27	10	55
	testing.mp3	4.35	331	30	10	14			
	advacedFile.mp3	2.84	332	17	8	8			
	ed_sheeran.mp3	5.48	121	32	9	8			
	Taylorswift.mp3	4.89	322	19	9	10			
zip	FDSC.zip	77.2	331	152	7	135	71.6	5.6	60.8
	Frontend.zip	63.0	311	91	6	89			
	Backend.zip	28.5	133	38	4	32			
	MyProject.zip	38.4	331	62	4	40			
	Favicon_io.zip	0.18	3211	15	7	8			
pptx	CS05_M2.pptx	2.21	313	26	1	6	15.4	2	10
	Ppt_test.pptx	2.89	311	11	2	15			
	Ppt_test(1).pptx	2.89	133	8	2	12			
	M4_ppt.pptx	1.78	232	15	2	4			
	WNA_final.pptx	2.29	211	17	3	13			
docx	WMS.docx	4.91	333	21	7	15	15.2	3.8	10.6
	AK_assign_2.docx	2.67	333	18	3	11			
	AK_assign_4.docx	2.98	331	24	5	14			
	DIP_lab9.docx	0.3	111	9	3	7			
	Buglist.docx	0.1	3333	4	1	6			
pdf	WMS_diag.pdf	1.11	111	19	1	15	17.6	3.4	25.4
	Assignment1.pdf	0.99	333	16	7	39			
	CS457_assign.pdf	0.99	3333	14	3	12			
	Lab09_DIP.pdf	0.64	1111	21	2	34			
	AK_assign.pdf	0.89	3111	18	4	27			
csv	Dummydata.csv	0.03	111	13	7	6	13.2	5.2	4.4
	Stationdata.csv	0.18	1111	12	7	6			
	Dataset.csv	0.07	333	13	4	5			
	Station-data1.csv	0.03	111	12	6	3			
	Ts-test.csv	0.02	1111	16	2	2			
txt	Test.txt	0.0001	113	5	2	1	13.2	3.6	3.4
	groupMember.txt	0.0001	333	13	1	2			
	Test(1).txt	0.0001	3232	18	4	4			
	User_code.txt	0.0001	123	13	6	3			
	SQl_script.txt	0.0012	3333	17	5	7			
exe	Tahometrx89.exe	5.35	121	33	13	24	47.4	16.6	20.4
	VSCodeSetup.exe	90.97	333	87	19	48			
	Python3.11.3.exe	24.8	1111	58	26	14			
	EAppInstaller.exe	12.7	113	42	21	9			
	ChromeSetup.exe	1.35	331	17	4	7			
jpg	Cover1.jpg	0.08	111	13	1	4	11	1	2.8
	Cover4.jpg	0.08	111	12	1	3			
	ERD.jpg	0.06	333	14	1	2			
	ERD2.jpg	0.06	3333	9	1	3			
	Plag_report.jpg	0.04	1331	7	1	2			

Table 2 shows the summary of test with different file types and sizes. It shows the time that passed for uploading, downloading, and deleting each file. File sizes are shown in megabytes, while duration for all these actions is expressed in seconds. Also, the average values for uploading, downloading, and deleting each file type are given. This data helps explain the relationship between file sizes and the time required for these operations in a more accessible way. From the experimental data, a clear pattern emerges. The results can be classified into three types depending on the file size and its type.

For smaller file size (such as .txt and .csv) require significantly less time for all operations, typically completing uploads, downloads, and deletions in just a few seconds. Medium-sized files (such as .pdf, .docx, and .pptx) show moderate variations, with uploads taking slightly longer compared to deletions and downloads. Larger files (such as .zip, .mp3, .exe) take considerably more time, particularly during the upload process. Notably, .zip files had the highest upload duration, often exceeding 60 seconds, while .exe files exhibited the most variation in time consumption. These results demonstrate a direct relationship between file size and the time taken for each operation. Uploading is consistently the most time-intensive process, followed by downloading, while deletion remains the quickest across all file types. These insights highlight the importance of optimizing file transfer processes, especially when handling large files.

Larger file sizes generally take longer to upload, with a steep increase for ZIP files, where larger files require more time to download. Deletion times are less dependent on file size, except for MP3 files, which take significantly longer.

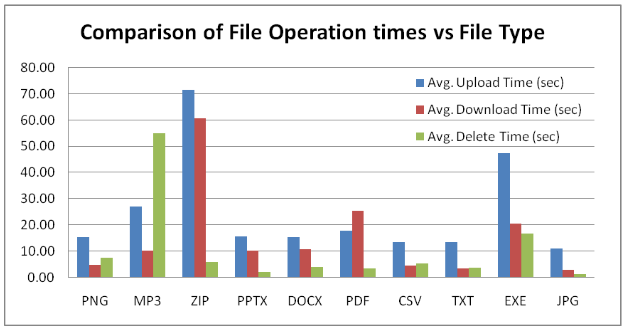

Figure 5: Showing File type vs file operation time

ZIP and EXE files requiring the most time for uploads and downloads. This is likely due to file integrity checks, encryption, and network latency. |It can be seen from Figure 5 that downloading is generally faster than uploading, especially for compressed and multimedia files, as they benefit from efficient retrieval mechanisms.
Deletion is the quickest operation, with most files being removed within a few seconds.

CONCLUSION

To enhance data security during transmission and in storage a valuable tool is designed. This newly designed system will protect data from cyber-attacks and ransom ware. It enhances security in three steps first by segmenting the data, and then encrypting the chunks and then saving it on distributed servers. The other features like user registration, file uploads, downloads, deletion can be performed with ease. Users can access their data flexibly and securely. This system makes a significant impact in the field of data security, providing strong data protection solutions to thwart cyber threats and maintain security, and authenticity of user.

The proposed system will share data within an organization in a dependable and effective way while giving priority to security and performance. In a nutshell, data partitioning can be used to enhance scalability by splitting data into smaller, manageable segments, which helps in lightening the burden on individual servers. Encryption is the most vital application in securing user data for file sharing and database management applications. With state of the art AES algorithm, data can be protected from theft by unauthorized persons.

REFERENCE

[1] Farooq, M., Younas, R. M. F., Qureshi, J. N., Haider, A., Nasim, F., & Khan, H. (2025). Cyber security Risks in DBMS: Strategies to Mitigate Data Security Threats: A Systematic Review. Spectrum of engineering sciences, 3(1), 268-290.

[2] Eleimat, M., & Őszi, A. (2025). Cybersecurity in Aviation: Exploring the Significance, Applications, and Challenges of Cybersecurity in the Aviation Sector. Periodica Polytechnica Transportation Engineering.

[3] Yang, Y., He, H., Feng, Z., Chen, F., & Yuan, Y. (2025). Cloud-Based Privacy-Preserving Medical Images Storage Scheme with Low Consumption. IEEE Transactions on Multimedia.

[4] Rijmen, V., & Daemen, J. (2001). Advanced encryption standard. Proceedings of federal I nformation processing standards publications, national institute of standards and technology, 19, 22.

[5] Gray, J., & Reuter, A. (1992). Transaction processing: concepts and techniques. Elsevier.

[6] GDPR, G. (2016). General data protection regulation. Regulation (EU), 679.

[7] Delfs, H., Knebl, H., & Knebl, H. (2002). Introduction to cryptography (Vol. 2). Heidelberg: Springer.

[8] Agrawal, E., & Pal, P. R. (2017). A secure and fast approach for encryption and decryption of message communication. Int. J. Eng. Sci, 11481.

[9] Rogaway, P. (2004, February). Nonce-based symmetric encryption. In International workshop on fast software encryption (pp. 348-358). Berlin, Heidelberg: Springer Berlin Heidelberg.

[10] Bellare, M., & Tackmann, B. (2016). Nonce-based cryptography: retaining security when randomness fails. In Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I 35 (pp. 729-757). Springer Berlin Heidelberg.

[11] Mahmud, M. S., Huang, J. Z., Salloum, S., Emara, T. Z., & Sadatdiynov, K. (2020). A survey of data partitioning and sampling methods to support big data analysis. Big Data Mining and Analytics, 3(2), 85-101.

[12] Ke, Q., Prabhakaran, V., Xie, Y., Yu, Y., Wu, J., & Yang, J. (2016). U.S. Patent No. 9,235,396. Washington, DC: U.S. Patent and Trademark Office.

[13] Nafi, K. W., Kar, T. S., Hoque, S. A., & Hashem, M. M. A. (2013). A newer user authentication, file encryption and distributed server based cloud computing security architecture. arXiv preprint arXiv:1303.0598.

[14] Sears, R., Van Ingen, C., & Gray, J. (2007). To blob or not to blob: Large object storage in a database or a filesystem?. arXiv preprint cs/0701168.

[15] MySQL, A. B. (2005). MySQL: the world's most popular open source database. http://www. mysql. com/.

[16] Singh, G. (2013). A study of encryption algorithms (RSA, DES, 3DES and AES) for information security. International Journal of Computer Applications, 67(19).

[17] De Los Reyes, E. M., Sison, A. M., & Medina, R. (2019). Modified AES cipher round and key schedule. Indonesian Journal of Electrical Engineering and Informatics (IJEEI), 7(1), 29-36.

[18] Chow, S., Eisen, P., Johnson, H., & Van Oorschot, P. C. (2003). White-box cryptography and an AES implementation. In Selected Areas in Cryptography: 9th Annual International Workshop, SAC 2002 St. John's, Newfoundland, Canada, August 15–16, 2002 Revised Papers 9 (pp. 250-270). Springer Berlin Heidelberg.

[19] Akkar, M. L., & Giraud, C. (2001). An implementation of DES and AES, secure against some attacks. In Cryptographic Hardware and Embedded Systems—CHES 2001: Third International Workshop Paris, France, May 14–16, 2001 Proceedings 3 (pp. 309-318). Springer Berlin Heidelberg.

[20] Ahmad, N., Wei, L. M., & Jabbar, M. H. (2018, June). Advanced Encryption Standard with Galois Counter Mode using Field Programmable Gate Array. In Journal of Physics: Conference Series (Vol. 1019, No. 1, p. 012008). IOP Publishing.

[21] Käsper, E., & Schwabe, P. (2009, September). Faster and timing-attack resistant AES-GCM. In International Workshop on Cryptographic Hardware and Embedded Systems (pp. 1-17). Berlin, Heidelberg: Springer Berlin Heidelberg.

[22] Lapid, B., & Wool, A. (2019). Cache-attacks on the ARM TrustZone implementations of AES-256 and AES-256-GCM via GPU-based analysis. In Selected Areas in Cryptography–SAC 2018: 25th International Conference, Calgary, AB, Canada, August 15–17, 2018, Revised Selected Papers 25 (pp. 235-256). Springer International Publishing.